04860.P3242

UNITED STATES PATENT APPLICATION

FOR

DECODING CODED DATA STREAMS

INVENTOR:

FELIX UWE BERTRAM

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 WILSHIRE BOULEVARD
SEVENTH FLOOR
LOS ANGELES, CA 90025-1026

(408) 720-8300

DECODING CODED DATA STREAMS

BACKGROUND

[0001]   The present invention relates to a method and apparatus for decoding coded data streams, including biphase coded data streams such as, similar to or compatible with the IEC 60958 and Japanese EIAJ CP-340 1987-9 standards. AES-EBU (Audio Engineering Society – European Broadcasting Union) and SPDIF (Sony Philips Digital Interconnect Format) are both implementations of the IEC 60958 standard.

[0002]   Each of these standards is now widely used for digital audio transmission. These standards were developed following the introduction of the compact disc player in which digital audio data signals were initially retained inside the player cabinet and were converted to analogue signals before leaving the cabinet. However, audio (and now video) signals are increasingly maintained in their digital format for as long as possible to maintain digital quality. Thus, it is now common for digital audio and video data source devices, such as music synthesisers and CD, DVD, DAT and MP3 players to output the audio and video data in digital form to subsequent audio and video logic circuits. These logic circuits may be implemented in separate devices, such as desktop computers, amplifiers and televisions. The IEC 60958 standard was developed to aid transfer of digital signals between devices and has now effectively become the worldwide standard.

[0003]   In common with other standards, the IEC 60958 standard uses biphase coding, such as biphase Manchester coding. In biphase coding, a data stream is combined with a clock on a single channel such that there are up to two transitions on the line for each data bit conveyed. Specifically, with biphase Manchester coding, there is a line transition at the end of each data bit period and a central transition if the bit is a '1'.

[0004]   An example of biphase coding is shown in Fig. 1. Specifically, Fig. 1 shows an uncoded data signal in the middle, the data signal including a plurality of data bits. The clock

for this data signal is shown above the data signal. The clock toggles or transitions from lo to hi at the start of each bit in the data signal. Thus, the clock includes a rising edge for every data bit. The corresponding biphase coded data is shown at the bottom of Fig. 1. Every data bit of the data signal is represented as two states in the biphase coded data, which each form a cell. Thus, the length of two cells is equal to the length of a data bit. Accordingly, where all flip-flops toggle on the same clock edge (for example, the rising edge), the clock required to produce the coded signal must have twice the frequency of the clock shown at the top of Fig. 1. As an alternative, it would be possible to use both clock edges to achieve the same effect. Such a clock is termed double data rate or double pumped. What is important, is that the clock provides two triggering events per data bit.

[0005]   Since there is a transition at the end of each bit period in the biphase coded signal, the logical level at the start of a data bit is always inverted to the level at the end of the previous bit. In other words, the first cell corresponding to a bit in the biphase coded signal will always have a different state to that of the second cell of the preceding bit. Moreover, since there is a central transition if the bit is a '1', the level at the end of a bit is equal (a '0' transmitted) or inverted (a '1' transmitted) to the start of the bit. This is sometimes termed biphase mark coding (BMC).

[0006]   In the IEC 60958 standard, each audio frame contains two sub-frames. Each subframe contains one audio sample word. Such an audio sample word is a 32 bit word. In the IEC 60958 standard, the 32 bits in a subframe may be used as follows:

Bit	Meaning
0-3	Preamble
4-7	Auxiliary audio data bits
8-27	Sample
28	Validity

29   Subcode data

30   Channel status information

31   Parity

[0007]   Depending on the technology, different sample sizes are used in subframes. Thus, a 24 bit sample using bits 4-27 can be obtained. In contrast, a CD player uses only 16 bits so only bits 13(LSB)-27(MSB) are used. Bits 4-12 are set to '0'. However, it should be remembered that, when encoded, each '0' bit includes a transition.

[0008]   The number of subframes that is used depends on the number of channels that is transmitted, although no scheme is yet known that uses more than two subframes. If two channels are transmitted, then each frame contains two subframes and each of the two subframes contains a sample. The two subframe samples are played simultaneously on the respective channels. A block contains 192 subframes, numbered 0-191.

[0009]   The sampling frequency (fs) is the frequency of samples in the data signal. Since two 32 bit words are transmitted for each sample, 64 bits are transmitted for each sample. Accordingly, the frequency of the clock shown in Fig. 1 is 64 times the sampling frequency (64fs). Since two cells of are provided in the biphase coded signal for each data bit, the biphase coded signal clock frequency is 128 times the sampling frequency (128fs).

[0010]   Different technologies use different sampling frequencies. For example, for CD fs=44.1kHz and for DAT fs=48kHz. Accordingly, for CD 128fs=5.6MHz and for DAT fs=6.1MHz. Thus, a block containing 192 subframes typically accounts for approximately 4ms of audio (192/44.1kHz=4.3ms and 192/48kHz=4ms).

[0011]   There are several technologies available for implementing digital to analogue converters (DACs), including R-2R ladders and constant calibration DACs. R-2R ladders are created from a network of precision resistors. Constant calibration DACs generate an output

voltage by combining the charges of several capacitors, which are constantly re-calibrated. All DAC technologies have in common that they are expensive to manufacture. Accordingly, typical audio converters include DACs with less physical resolution than the 24 bit words containing the audio samples. Thus, to achieve the required resolution, an oversampling technique is used. In other words, the data is sampled at a multiple of the required sampling frequency fs and then filtered. These oversampling mechanisms are driven using a faster clock – that is, one running at a higher frequency. The audio clock of most audio converters therefore runs at 256 times the sampling frequency (256fs), although higher frequencies such as 384fs are known. As the received IEC 60958 signal carries the sampling clock fs in itself, the oversampling audio clocks are phase locked to the frequencies shown in Fig. 1.

[0012]   As noted above, the first four bits (0-3) in each subframe form a preamble or header. The preamble thus contains eight cells and feasibly up to 8 transitions. The preamble is used for synchronisation and does not carry any audio data. Consequently, it does not use biphase coding. Thus, the preamble is capable of including more than two 0s or 1s in succession and always does so at the start to distinguish itself from the rest of the subframe. In practice then, the preamble never contains the 8 possible transitions.

[0013]   In the IEC 60958 standard, the preamble forms one of three allowed synchronisation patterns, as follows:

Preamble	Cell Order	
	Last Cell '0'	Last Cell '1'
B	11101000	00010111
M	11100010	00011101
W	11100100	00011011

Preamble B marks a word containing data for channel A (left channel) at the start of the data block; preamble M marks a word containing data for channel A that is not at the start of the

data block; and preamble W marks a word containing data for channel B (right channel), or any other channel except A.

[0014] Typical methods of decoding biphase coded data streams involve the use of analogue or digital phase locked loops (PLLs) to recover the biphase coded signal clock, which is at twice the bit rate or 128fs. Typically, both types of PLL include, for example, a voltage controlled oscillator (VCO) generating a clock having approximately 128fs. Transitions in the data stream are detected and low frequency transitions are filtered out so that only transitions at the approximate sampling frequency are retained. These transitions are used to adjust the phase and frequency of the clock generated by the PLL so that it matches the data clock of the written data. Thus, if the phase or frequency of written data changes, the PLL recognises these changes and adjusts accordingly. Once this data clock is recovered, it can be used to determine whether a transition has occurred between PLL clock cycles and hence to recover the proper stream of 0s and 1s in the encoded data. The encoded data can then be decoded using a reverse algorithm to the coding algorithm.

[0015] However, analogue PLLs suffer the drawback that they cannot easily be integrated with digital logic in typical application specific integrated circuits (ASICs), gate arrays and field programmable gate arrays (FPGAs). Digital solutions are therefore preferred. Unfortunately, digital PLLs also suffer a number of drawbacks, as discussed below.

[0016] The recovered IEC 60958 128fs data clock rate is phase locked to the sampling frequency fs carried by the IEC 60958 signal. In addition, the oversampling frequency 256fs for use in the audio converter must be created from the recovered IEC 60958 data clock. Thus, any jitter in the sampling frequency fs will degrade audio performance since aperture jitter (variations in the window during which a signal is sampled), bit resolution and signal-to-noise are inter-related. Clock jitter therefore audibly (or visibly) degrades the output of the audio (or video) output device. Typically, the AES/EBU standard for serial digital audio uses

a 163ns clock rate (or 6.1MHz) and allows up to +/- 20ns of jitter in the signal (equivalent to 50MHz). Other standards are more stringent. Consequently, the jitter on the PLL data clock must be minimised to preserve audio fidelity. In other words, the window in which each rising edge of the clock transition may be generated must be minimised.

[0017]   Improvements to biphase and other signal decoding therefore present a number of problems. First, it is desired to create a 256fs clock with very little jitter to feed the audio (or other) system. In addition, it is desired to use the same clock to decode the incoming biphase coded data stream, such as an IEC 60958 or other coded signal, since this signal is phase locked with the signal output to the audio system. Thus, it must be possible to adjust this clock fast enough to track the phase jitter present in the biphase coded data stream, so that the bits can be consistently and accurately decoded. Moreover, the phase of the PLL must be set so that the centre of each cell in the biphase coded data stream is sampled. A fast PLL capable of tracking the IEC 60958 signal will have a wide lock range, but create a lot of jitter. In contrast, a stable PLL with low jitter will have a narrow lock range, but may not be fast enough to track the incoming IEC 60958 signal.

[0018]   The usual approach for implementing the PLL clock would entail providing a fast free-running clock; using a programmable divider with a very high resolution; adjusting the programmable divider so that it follows or tracks the preambles; and using the (appropriately divided) output of the divider to decode the cells.

[0019]   Unfortunately, there are several problems with this approach. In particular, this approach requires the phase reference to be determined by detecting preambles using a circuit running from the free-running clock. This means that the minimum obtainable error in the phase reference is dependent on the frequency of the free-running clock. More specifically, the minimum obtainable error decreases as the speed of the free-running clock increases. This phase reference error will effect the accuracy of the PLL and hence its ability to sample

the centre of each cell in the biphase coded data stream. Thus, the accuracy of the clock generated by the PLL is limited by the accuracy of the phase reference and hence by the frequency of the free-running clock.

[0020]   In addition, a digital PLL is a regulator, which controls the frequency of a programmable clock generator depending on the output of a phase comparator, which compares the phase of the generated signal with a phase reference. In this case, the phase reference is the phase of the detected preambles in the biphase coded signal. However, a simple P-type regulator will not be capable of tracking the digital signal with the required accuracy for this application.

[0021]   It might be feasible to obtain the required accuracy using a PID-type regulator. Unfortunately, however, PID-type regulators introduce additional problems. In particular, PID-type regulators include arithmetic to control the clock signal. However, such arithmetic is slow and large when implemented in digital logic. In addition, there is a fixed relationship between the lock range and the accuracy of PLLs, which is dependent on the loop filter of the PLL. Thus, it is difficult to track different frequencies (for example, 44.1kHz, 88.2kHz etc for CD; and 48kHz, 96kHz etc for DAT) using a single set of parameters. Switching between different parameter sets requires further circuitry and is costly. Accordingly, implementation of the PLL using a PID-type regulator would be difficult at the required clock frequencies.

[0022]   In summary, depending on the technology in which the apparatus is to be implemented, a PLL with the required clock speeds and accuracy is difficult to implement in a cost effective manner.

SUMMARY

[0023]   According to certain embodiments of the present invention, there is provided a method of decoding a coded data stream of a plurality of cells, the uncoded data stream comprising a plurality of segments each formed of a preamble of fixed length and a series of data bits, wherein:

the coded data stream includes at least two cells and one transition for each data bit and a portion without a transition at the beginning of each preamble, the said portion having the maximum period in the coded data stream without a transition and being longer than a bit period, the method characterised by:

detecting transitions in the coded data stream;

determining the maximum period in the data stream without a transition;

determining a first threshold corresponding to a period of time less than the maximum period and greater than the data bit period;

determining whether a considered transition corresponds to a data bit or a preamble based on the time between the considered transition and a preceding transition compared with said first threshold; and

if the considered transition corresponds to a data bit, determining the data bit based on the considered transition and the preceding considered transition.

[0024]   According to another aspect of the present invention, there is provided an apparatus for decoding a coded data stream of a plurality of cells, the uncoded data stream comprising a plurality of segments each formed of a preamble of fixed length and a series of data bits, wherein:

the coded data stream includes at least two cells and one transition for each data bit and a portion without a transition at the beginning of each preamble, the said portion having the maximum period in the coded data stream without a transition and being longer than a bit period, the method characterised by:

a detector for detecting transitions in the coded data stream;

a timer for determining the time between adjacent transitions;

logic for determining the maximum period in the data stream without a transition and for determining a first threshold corresponding to a period of time less than the maximum period and greater than the data bit period; and

a filter for determining whether a considered transition corresponds to a data bit or a preamble based on the time between the considered transition and a preceding transition compared with said first threshold and, if the considered transition corresponds to a data bit, determining the data bit based on the considered transition and the preceding considered transition.

[0025]   According to yet another aspect of the present invention, there is provided a method of decoding a biphase coded data stream comprising a plurality of words formed of a plurality of cells of equal period, the cells in each word forming a preamble and a series of data bits, wherein:

the preamble comprises a predetermined plurality of cells and each data bit comprises two cells;

there is a transition between cells of adjacent data bits;

there is a transition between a data bit cell and an adjacent preamble cell; and

a predetermined maximum period in the data stream without a transition is formed at the beginning of the preamble, the maximum period being longer than the cell period, the method characterised by:

detecting transitions in the data stream;

determining a threshold corresponding to the maximum period in the data stream without a transition;

determining whether a transition corresponds to a data bit cell or the preamble based on the time between the transition and a preceding transition compared to the threshold; and

if the transition corresponds to a data bit cell, determining the data bit represented by the data bit cell based on the transition and the transition of a cell of an adjacent data bit.

[0026]   According to a still further aspect of the present invention, there is provided an apparatus for decoding a biphase coded data stream comprising a plurality of words formed

of a plurality of cells of equal period, the cells in each word forming a preamble and a series of data bits, wherein:

the preamble comprises a predetermined plurality of cells and each data bit comprises two cells;

there is a transition between cells of adjacent data bits;

there is a transition between a data bit cell and an adjacent preamble cell; and

a predetermined maximum period in the data stream without a transition is formed at the beginning of the preamble, the maximum period being longer than the cell period, the apparatus comprising:

a detector for detecting transitions in the data stream;

logic for determining a threshold corresponding to the maximum period in the data stream without a transition; and

a filter for determining whether a transition corresponds to a data bit cell or the preamble based on the time between the transition and a preceding transition compared to the threshold and, if the transition corresponds to a data bit cell, for determining the data bit represented by the data bit cell based on the transition and the transition of a cell of an adjacent data bit.

BRIEF DESCRIPTION OF THE DRAWINGS

[0027]   Embodiments of the present invention will now be described by way of further example only and with reference to the accompanying drawings, in which:

[0028]   Fig. 1 illustrates the principles of biphase coding;

[0029]   Fig. 2 is a schematic view of a decoding apparatus according to an embodiment of the present invention;

[0030]   Fig. 3 is a state machine diagram illustrating a decoding method and apparatus according to an embodiment of the present invention;

[0031]   Figs. 4, 5 and 6 illustrate the decoding of exemplary data streams according to embodiments of the present invention;

[0032] Figs. 7, 8 and 9 are state machine diagrams illustrating other decoding methods and apparatuses according to embodiments of the present invention;

[0033] Fig. 10 is an illustration of an enable signal output by the apparatus of the present invention; and

[0034] Fig. 11 is a schematic illustration of a clock recovery apparatus according to an embodiment of the present invention.

DETAILED DESCRIPTION

[0035] As shown in Fig. 2, the data stream decoder 100 comprises a clock generator 10, a sampler 30, an edge detector 40, a distance counter 50 and a pulse filter 60. The clock generator 10 generates a free running clock with a clock frequency that is significantly higher than the sampling frequency fs. For example the clock generator 10 may generate a clock with a frequency of 96MHz for SPDIF signals. The free running clock is used to detect the state of the incoming data stream 20 using sampler 30. The synchronous edge detector 40 creates a pulse whenever the signal of the incoming data stream toggles – that is, whenever a transition is detected. Accordingly, in contrast to the prior art, a PLL is not used to detect transitions or their timing.

[0036] Pulses from the edge detector 40 are provided to the distance counter 50 and the pulse filter 60 separately. The distance counter measures the distance (or more correctly, the time) between pulses output by the edge detector 40 and, consequently, the distance or time between transitions in the incoming encoded data stream 20. From this, the maximum distance $dmax$ between two pulses and hence the maximum distance or time between transitions in the incoming data signal 20 is determined. Based on the characteristics of the encoded data stream, this distance $dmax$ is used to determine a number of thresholds for decoding the data stream 20.

[0037] More specifically, for biphase coding the maximum distance *dmax* between two pulses output by the edge detector always occurs at the start of the preamble coding and corresponds with three cells each having the same state. This is clear from the three allowed synchronisation patterns B, M and W, which always commence with 000 or 111, depending on the state of the preceding cell. The end of the preamble is always five cells after the end of the third cell in the preamble (the preamble includes a total of eight cells) and is always followed by a cell corresponding to an uncoded data bit. Finally, every two biphase coded cells correspond to an uncoded data bit.

[0038] Thus, the decoder 100 needs to determine two or more of the following distances, with reference to the sampling frequency, in order to set appropriate thresholds:

- 2x(128fs): the distance between the beginning of the first cell and the end of the second cell corresponding to a data bit – that is, the distance taken up by two cells. This distance and the corresponding threshold (hereinafter the 2cell threshold) is used to skip the transition in the middle of each '1' bit and to jump from bit to bit;

- 3x(128fs): the distance taken up by three cells. This distance and the corresponding threshold (hereinafter the 3cell threshold) is used to detect the preamble coding which violates the biphase coding scheme since it includes more than two 0s or 1s in a row at the start; and

- 5x(128fs): the distance taken up by five cells. This distance and the corresponding threshold (hereinafter the 5cell threshold) is used to find the end of the preamble after detecting the 3x(128fs) protocol violation.

[0039]   Since one cell length=$dmax$/3, appropriate threshold values may be calculated efficiently using simple digital logic without the use of multipliers as follows:

2cell threshold = 1.5 x cell length = 0.5$dmax$

3cell threshold = 2.25 x cell length = 0.75$dmax$

5cell threshold = 4.5 x cell length = 1.5$dmax$

Once these thresholds have been calculated, the distance counter 50 and the pulse filter 60 can be used to decode the data stream.

[0040]   In common with the calculation of the thresholds, the functionality of the pulse filter 60 is based on the encoding of the incoming data stream, which (as explained above) defines a maximum distance between transitions for a data bit. To achieve the required functionality, the pulse filter 60 may be constructed as a simple state machine, although other constructions will be obvious to persons skilled in the art. In particular, the pulse filter 60 can be implemented using software.

[0041]   One embodiment of the process carried out by a suitable pulse filter state machine 200 for biphase decoding is shown in Fig. 3. As shown in Fig. 3, the state machine 200 defines "holdoff" periods during which transition pulses from the edge detector 40 are ignored according to the current state and the distance from the last pulse received.

[0042]   Accordingly, as shown at 210, the pulse filter 60 first waits for the next pulse from the edge detector 40. When the next pulse is received, the pulse filter receives a determination from the distance counter 50 as to the distance between that pulse and the preceding pulse. If the distance is greater than the 3cell threshold (which is 2.25 cells long), then a biphase coding violation has occurred and the start of a preamble is detected, as shown at 260. Subsequently, the holdoff period is set at the 2cell threshold (which is 1.5 cell

lengths), as shown at 270, and transitions are ignored until the holdoff period has ended, as shown at 280, 290. All transitions for 1.5 cell lengths are therefore ignored. This is not problematic because, as explained above, the preamble does not contain audio data.

[0043]   Once the holdoff period has expired, the pulse filter 60 waits for the next two pulses from the edge detector 40, as shown at 300. The distance between the next two transitions in both M and W preambles is one cell length, whereas the distance between the next two transitions in B preambles is 3 cell lengths. Accordingly, if the distance detector 50 detects that the distance between the next two pulses is less than the 3cell threshold (that is, less than 2.25 cell lengths) an M or W preamble is detected, as shown at 310. Subsequently, the pulse filter 60 waits for the next pulse, as shown at 320, which will be occasioned by the transition between the end of the preamble and the next data bit cell. Accordingly, the end of the M or W preamble is detected 330. When this transition is detected, the holdoff period is set to the 2cell threshold 340 and further transitions are ignored until the holdoff period of 1.5 cell lengths has ended (see 240 and 250). The state machine then waits for the next pulse, as indicated at 210.

[0044]   If a preamble is detected at 260 and the distance between the next two detected pulses (as shown at 300) is greater than the 3cell threshold, then a B preamble is detected at 350. In that case, the last pulse to have been detected will correspond to the transition between the end of the B preamble and the next data bit. The holdoff period is therefore set to the 2cell threshold 360 and further transitions are ignored until the holdoff period of 1.5 cell lengths has ended (see 240 and 250). The state machine then waits for the next pulse, as indicated at 210, which should correspond to a data bit.

[0045]   If the end of the preamble has been previously detected, the next pulse after the end of the holdoff period should correspond to the transition between cells at the end of a data bit. Consequently, the distance between the last two detected pulses should not be

greater than the 3cell threshold. Accordingly, if the pulse filter 60 detects that the distance between the two pulses is not greater than the 3cell threshold, a data bit is detected. This is indicated at 220 in Fig. 3. In that case, the holdoff period is set to the 2cell threshold (as shown at 230). At the same time, the pulse filter 60 determines whether the "transition direction" of the most recently detected transition in the biphase coded signal is the same as the "transition direction" of the previously detected transition. Thus, if two rising edges are detected, then a '1' is output. Similarly, if two falling edges are detected, then a '1' is again output. However, if a rising edge is detected followed by a falling edge, or a falling edge is detected followed by a rising edge, then a '0' is output. Decoding in this manner is possible since all transitions in the middle of '1' data bits have been ignored.

[0046] The state machine then waits until the 2cell holdoff period has expired (as shown at 240) before receiving the next pulse (as shown at 210).

[0047] The decoding of an exemplary coded data stream 20 by the decoder 100 using the pulse filter 200 is shown in Fig. 4. Specifically, the figure shows a portion 500 of a data stream comprising a B preamble 550 and a series of uncoded data bits 560. Directly below data stream portion 500, the encoded data 520 is shown. Only the data bits 560 have been encoded using biphase coding. Thus, each encoded data bit comprises two cells. At least one of the two cells in each data bit transitions. Where there is a transition between adjacent cells, this is indicated T_y below the cells. Thus, the edge detector will output a pulse at each location T_y.

[0048] The first distance between transitions that the decoder is able to detect is distance 'a' between transitions T_1 and T_2. Since (a>3cell threshold), a preamble is detected and the holdoff period is set at the 2cell threshold. This means that all transitions for the period corresponding to 1.5 cell lengths following T_2 are ignored. Hence, transition T_3 is ignored. Thus, the next two transitions to be considered by the pulse filter 60 are transitions T_4 and T_5.

The distance between T_5 and the preceding transition T_4 is 'b'. Since (b>3cell threshold), the preamble is determined to be a B preamble. Accordingly, the holdoff period is again set at the 2cell threshold.

[0049]   This means that all transitions for the period corresponding to 1.5 cell lengths following T_5 are ignored. Thus, transition T_6 is ignored by pulse filter 60, although not by distance counter 50. Thus, the next transition to be considered by the pulse filter 60 is transition T_7. The pulse filter 60 therefore obtains the distance between transition T_7 and the preceding transition from the distance counter 50. The distance between T_7 and the preceding transition T_6 is 'c'. Since (c<3cell threshold), a data bit is detected and the holdoff period is set at the 2cell threshold. Moreover, since transitions T_5 and T_7 are both falling edges, the decoder outputs a data bit '1'.

[0050]   Because the holdoff period is set at the 2cell threshold (which means that all transitions for the period corresponding to 1.5 cell lengths following T_7 are ignored), transition T_8 is ignored. Thus, the next transition to be considered by the pulse filter 60 is transition T_9. The distance between T_9 and the preceding transition T_8 is 'd'. Since (d<3cell threshold), a data bit is detected and the holdoff period is set at the 2cell threshold. Moreover, since two falling edges are considered by the pulse filter 60, the decoder outputs a data bit '1'.

[0051]   Because the holdoff period is set at the 2cell threshold, all transitions for the period corresponding to 1.5 cell lengths following T_9 are ignored. In this case, there are no transitions in this holdoff period. The next transition to be considered by the pulse filter 60 is transition T_{10}. The distance between T_{10} and the preceding transition T_9 is 'e'. Again, (e<3cell threshold), a data bit is detected and the holdoff period is set at the 2cell threshold. Moreover, since the pulse filter 60 considers a falling edge followed by a rising edge, the decoder outputs a data bit '0'.

[0052] The decoder carries on detecting data bits until a further preamble is detected. This is illustrated in the portion of the data stream 520 shown in Fig. 5, which includes two regions of uncoded bits 560 that sandwich an M preamble 570. In common with Fig. 4, the pulse filter 60 will consider distances 'g', 'h' and 'i' in the initial portion of the data stream 560 and output bits '1', '0' and '1' respectively. Next, the pulse filter 60 will detect that the distance 'j' between transitions T_{25} and T_{26} is greater than the 3cell threshold and therefore detect a preamble. Subsequently, the holdoff period is set at the 2cell threshold so that all transitions for the following 1.5 cell lengths are ignored. In this case, no transitions occur within this holdoff period. Thus, the next two transitions detected at 300 in the state machine diagram of Fig. 3 are T_{27} and T_{28}. The distance 'k' between the two is less than the 3cell threshold (k<3cell threshold), so an M or W preamble is detected.

[0053] Subsequently, the pulse filter 60 waits for the next pulse, which is generated for transition T_{29}. This transition corresponds with the end of the M preamble. The holdoff period is set at the 2cell threshold, which means that all transitions for the period corresponding to 1.5 cell lengths following T_{29} are ignored by the pulse filter. Thus, transition T_{30} is ignored. The next transition to be considered by the pulse filter 60 is therefore transition T_{31}. The distance between T_{31} and the preceding transition T_{30} is 'l'. Since (l<3cell threshold), a data bit is detected and the holdoff period is set at the 2cell threshold. Moreover, since two falling edges are detected, the decoder outputs a data bit '1'. Again, the decoder carries on detecting data bits until a further preamble is detected.

[0054] The detection of a W preamble is now explained with reference to Fig. 6. Again, the pulse filter detects a preamble at 260 in the state machine diagram of Fig. 3 since the distance 'n' between transitions T_{40} and T_{41} is three cells so (n>3cell threshold). Subsequently, the holdoff period is set at the 2cell threshold so that all transitions for the following 1.5 cell lengths are ignored. In this case, no transitions occur within this holdoff

period. Thus, the next two transitions detected at 300 in the state machine diagram of Fig. 3 are T_{42} and T_{43}. The distance 'p' between these two transitions is less than the 3cell threshold (p<3cell threshold), so an M or W preamble is detected.

[0055]   Subsequently, the pulse filter 60 waits for the next pulse, at transition T_{44}, which corresponds with the end of the W preamble, and sets the holdoff period at the 2cell threshold. This means that all transitions for the period corresponding to 1.5 cell lengths following T_{44} are ignored. Again, the decoder then detects data bits until a further preamble is detected.

[0056]   Of course, other implementations of the pulse filter can be contemplated. These may depend on the information required to be obtained from the decoded signal. For example, where biphase coding is used but it is not desired to distinguish between different kinds of preamble, then a pulse filter 60 implemented as a state machine 600 as shown in Fig. 7 could be used. In this implementation, wait for next pulse 210, detect data bit 220, set holdoff period 230, wait for holdoff period 240 and decrement 250 are the same as in Fig. 3. In addition, detect preamble 610 of Fig. 7 corresponds with detect preamble 260 in Fig. 3. Thus, if a gap between detected signals exceeds the 3cell threshold, a preamble is again detected.

[0057]   However, in this instance it is not desired to distinguish between different types of preamble. Accordingly, once a preamble is detected, the holdoff period is set to the 5cell threshold, as shown at 620. The pulse filter then waits (as shown at 630), for example by decrementing a counter (as shown at 640), until the holdoff period has expired. The pulse filter 60 then waits for the next pulse (as shown at 650), which corresponds to the transition between the end of the preamble and the next data bit. The holdoff period is then set at the 2cell threshold (as shown at 660), before proceeding to wait for the holdoff period to expire (as shown at 240 and 250) and thence to wait for the next pulse (as shown at 210) as before.

[0058]　　The principle of this implementation of the pulse filter will now be described with reference to Fig. 4. The first distance between transitions that the decoder is able to detect is distance 'a' between transitions T_1 and T_2. Since (a>3cell threshold), a preamble is detected and the holdoff period is set at the 5cell threshold. This means that all transitions for the period corresponding to 4.5 cell lengths following T_2 are ignored. Thus, transitions T_3 and T_4 are ignored. Thus, the next transition to be considered by the pulse filter 60 is transition T_5. The holdoff period is then set at the 2cell threshold again.

[0059]　　This means that all transitions for the period corresponding to 1.5 cell lengths following T_5 are ignored. Thus, transition T_6 is ignored and the next transition to be considered by the pulse filter 60 is transition T_7. The distance between T_7 and the preceding transition T_6 is 'c'. Since (c<3cell threshold), a data bit is detected and the holdoff period is set at the 2cell threshold. Moreover, since transitions T_5 and T_7 are both falling edges, the decoder outputs a data bit '1'. The decoder continues to decode the data stream in the same manner as before.

[0060]　　It will be clear that this implementation of the pulse filter operates on the M and W preambles in a similar way. In particular, all transitions between the code violation at the beginning of the preamble are ignored and the pulse filter is then set to detect data bits by skipping transitions occurring in the middle of those data bits.

[0061]　　Another possible implementation of the pulse filter 60 is shown in Fig. 8. In common with the implementation shown in Fig. 3, this implementation is able to distinguish between the B and M or W preambles. However, it uses a flag (fs1) to assist in distinguishing between preambles. At the start of operation, the flag is set at fs1=0.

[0062]   In common with the implementations described above, if the period between adjacent pulses is detected as being greater than the 3cell threshold, a preamble is detected at 610. If the test fs1=1 is false, then the state machine determines that the preamble can be any one of a B, M or W pulse (see 710). At the same time, the holdoff period is set to the 5cell threshold (see 715) and the flag is set fs1=1. The pulse machine then proceeds to wait for the holdoff period to expire (see 240 and 250) before again waiting for the next pulse (see 210).

[0063]   On the other hand, if fs1=1 when the preamble is detected at 610, then it is determined that the preamble is a B preamble. The holdoff period is then set to the 2cell threshold and the flag is set fs1=0, before again proceeding to wait for the holdoff period to expire (see 240 and 250) before waiting for the next pulse (see 210).

[0064]   If the period between adjacent cells is less than the 3cell threshold, then the pulse filter 700 determines at 720 either that a data bit has been detected or an M or W preamble has been detected, depending on whether the flag is set fs1=1 or not. If an M or W preamble is detected (see 730), the holdoff period is then set to the 2cell threshold and the flag is set fs1=0 (see 734), before again proceeding to wait for the holdoff period to expire (see 240 and 250) before waiting for the next pulse (see 210).

[0065]   On the other hand, if the test fs1=1 is false when the data bit or M or W preamble is detected at 720, then it is determined that a bit is detected (see 220). The holdoff period is then set to the 2cell threshold (see 230), before again proceeding to wait for the holdoff period to expire (see 240 and 250) and to wait for the next pulse (see 210).

[0066]   The principle of this implementation of the pulse filter will now be explained with reference to Fig. 4. The first distance between transitions that the decoder is able to detect is distance 'a' between transitions T_1 and T_2. Since (a>3cell threshold), a preamble is detected (see 610). Since the flag is set fs1=0, it is determined that the preamble could be any one of a

B, M or W preamble (see 710). The flag is then set fs1=1 and the holdoff period is set at the 5cell threshold (see 715). This means that all transitions for the period corresponding to 4.5 cell lengths following T_2 are ignored. Thus, transitions T_3 and T_4 are ignored.

[0067]   Thus, the next transition to be considered by the pulse filter 60 is transition T_5. The distance between T_5 and the preceding transition T_4 is 'b'. Since (b>3cell threshold) and the flag is set fs1=1, the preamble is determined to be a B preamble (see 610 and 750). Accordingly, the holdoff period is again set at the 2cell threshold and the flag is reset fs1=0 (see 760).

[0068]   This means that all transitions for the period corresponding to 1.5 cell lengths following T_5 are ignored. Thus, transition T_6 is ignored by pulse filter 60. Thus, the next transition to be considered by the pulse filter 60 is transition T_7. The distance between T_7 and the preceding transition T_6 is 'c'. Since (c<3cell threshold), the pulse filter 700 determines that either a data bit or an M or W preamble is detected (see 720). Since fs1=0, a data bit is detected (see 220) and the holdoff period is set at the 2cell threshold (see 230). Moreover, since transitions T_5 and T_7 are both falling edges, the decoder outputs a data bit '1'.

[0069]   The decoder carries on detecting data bits until a further preamble is detected. This is illustrated in the portion of the data stream 520 shown in Fig. 5. In common with Fig. 4, the pulse filter 60 will consider distances 'g', 'h' and 'i' in the initial portion of the data stream 560 and output bits '1', '0' and '1' respectively. Next, the decoder will detect that the distance 'j' between transitions T_{25} and T_{26} is greater than the 3cell threshold and therefore detect a preamble (see 610). Since the flag is set fs1=0, it is determined that the preamble could be any one of a B, M or W preamble (see 710). The flag is then set fs1=1 and the holdoff period is set at the 5cell threshold (see 715). This means that all transitions for the period corresponding to 4.5 cell lengths following T_{26} are ignored. Thus, transitions T_{27} and T_{28} are ignored.

[0070] Thus, the next transition to be considered by the pulse filter 700 at state 210 is transition T_{29}. The distance between T_{29} and the preceding transition T_{28} is 'x'. Since (x<3cell threshold), the pulse filter 700 determines that either a data bit or an M or W preamble is detected (see 720). Since fs1=1, the decoder determines that the preamble is an M or W preamble (see 730). Subsequently, the holdoff period is set at the 2cell threshold and the flag is reset fs1=0 (see 740). The decoder then proceeds to detect data bits as normal. It will be clear that the pulse filter 700 operates in a similar fashion on the W preamble shown in Fig. 6.

[0071] Although the implementations of the pulse filter 60 shown in Figs. 3, 7 and 8 are fully operational, the preferred and more efficient implementation is shown in Fig. 9. In common with the implementations described above, this implementation is able to distinguish between the B and M or W preambles. Again, a flag (fs1) is used to distinguish between B preambles on one hand and M or W preambles on the other. At the start of operation, this flag is set at fs1=0.

[0072] In common with the previous implementation, if the period between adjacent pulses is detected as being greater than the 3cell threshold, the existence of a preamble is detected at 810. If the test fs1=1 is false, then the state machine determines that the preamble can be any one of a B, M or W pulse (see 820). At the same time, the holdoff period is set to the 5cell threshold and the flag is set fs1=1 (see 830). The pulse machine then proceeds to wait for the holdoff period to expire (see 240 and 250) before again waiting for the next pulse (see 210).

[0073] On the other hand, if fs1=1 when the preamble is detected at 810, then it is determined that the preamble is a B preamble and a data bit is detected (see 850). The holdoff period is then set to the 2cell threshold and the flag is set fs1=0 (see 860), before

again proceeding to wait for the holdoff period to expire (see 240 and 250) and to wait for the next pulse (see 210).

[0074]   If the period between adjacent cells is less than the 3cell threshold, then the pulse filter 800 determines at 220 that a bit is detected. The holdoff period is then set to the 2cell threshold and the flag is set fs1=0 (see 840), before again proceeding to wait for the holdoff period to expire (see 240 and 250) and to wait for the next pulse (see 210).

[0075]   The principle of this implementation of the pulse filter will now be explained with reference to Fig. 4. The first distance between transitions that the decoder is able to detect is distance 'a' between transitions T_1 and T_2. Since (a>3cell threshold), a preamble is detected (see 810). Since the flag is set fs1=0, it is determined that the preamble could be any one of a B, M or W preamble (see 820). The flag is then set fs1=1 and the holdoff period is set at the 5cell threshold (see 830). This means that all transitions for the period corresponding to 4.5 cell lengths following T_2 are ignored. Thus, transitions T_3 and T_4 are ignored.

[0076]   Thus, the next transition to be considered by the pulse filter 60 is transition T_5. The distance between T_5 and the preceding transition T_4 is 'b'. Since (b>3cell threshold) and the flag is set fs1=1, the preamble is determined to be a B preamble (see 810 and 850). In addition, the pulse filter outputs a spurious data bit at the time of determining that the preamble is a B preamble. This spurious bit does not correspond with any of the data bits included in the original data signal 500 before biphase coding. The holdoff period is again set at the 2cell threshold and the flag is reset fs1=0 (see 860).

[0077]   This means that all transitions for the period corresponding to 1.5 cell lengths following T_5 are ignored. Thus, transition T_6 is ignored by pulse filter 60. Thus, the next transition to be considered by the pulse filter 60 is transition T_7. The distance between T_7 and the preceding transition T_6 is 'c'. Since (c<3cell threshold), a data bit is detected (see 220)

and the holdoff period is set at the 2cell threshold (see 230). Again, the flag is set fs1=0. Moreover, since transitions T_5 and T_7 are both falling edges, the decoder outputs a data bit '1'.

[0078] The decoder carries on detecting data bits until a further preamble is detected. This is illustrated in the portion of the data stream 520 shown in Fig. 5. In common with Fig. 4, the pulse filter 60 will consider distances 'g', 'h' and 'i' in the initial portion of the data stream 560 and output bits '1', '0' and '1' respectively. Next, the decoder will detect that the distance 'j' between transitions T_{25} and T_{26} is greater than the 3cell threshold and therefore detect a preamble (see 810). Since the flag is set fs1=0, it is determined that the preamble could be any one of a B, M or W preamble (see 820). The flag is then set fs1=1 and the holdoff period is set at the 5cell threshold (see 830). This means that all transitions for the period corresponding to 4.5 cell lengths following T_{26} are ignored. Thus, transitions T_{27} and T_{28} are ignored.

[0079] Thus, the next transition to be considered by the pulse filter 800 at state 210 is transition T_{29}. The distance between T_{29} and the preceding transition T_{28} is 'x'. Since (x<3cell threshold), the pulse filter 800 detects a data bit and the preamble is not determined to be a B preamble. However, this data bit is a spurious data bit and does not correspond with any of the data bits included in the original data signal 500 before biphase coding. Subsequently, the holdoff period is set at the 2cell threshold and the flag is reset fs1=0 (see 840). The decoder then proceeds to detect data bits as normal. It will be clear that the pulse filter 800 operates in a similar fashion on the W preamble shown in Fig. 6.

[0080] Accordingly, with this embodiment, the decoder outputs spurious data bits after each B, M and W preamble has been detected. However, since the sequencing of preambles and decoded data bits is known, it is a comparatively simple matter to remove these spurious data bits from the decoded data stream.

[0081]   It will be obvious to those skilled in the art that other implementations of the pulse filter can be used to obtain the desired results. Similarly, other ways of implementing the pulse filter will be readily apparent to persons skilled in the art without the exercise of inventive skill. For example, it would be possible to implement the pulse filter so that it is further capable of distinguishing between M and W preambles based on their different characteristics if desired. For example, it would be possible to distinguish M and W preambles based on the difference between the distance between transitions T_{26} and T_{27} in Fig. 5 on one hand and the distance between transitions T_{41} and T_{42} in Fig. 6 on the other.

[0082]   In addition, if the pulse filter is used for decoding a data stream that has been encoded other than by biphase coding, then it will be a relatively simple matter to redesign the pulse filter so that it is functional for the other coding system.

[0083]   In any event, the pulse filter 60 is able to output a fully decoded stream of digital audio or other data without the use of a PLL. Instead, the decoder 100 uses a free running clock generated by a stand alone digital clock generator 10. Moreover, since the first and last transitions sensed for all preambles always occur at the same place, the decoder (specifically, the pulse filter 60) is able to provide a consistent synchronisation signal 'sync' to indicate the beginning of an IEC 60958 block (where a B preamble is detected) or frame or subframe (where an M or W preamble is detected).

[0084]   It should be noted that in the IEC 60958 standard using two channels, data for the two channels is interleaved. Thus, subframes for channel A (indicated by preceding B and M preambles) are interleaved with subframes for channel B (indicated by preceding W preambles). Accordingly, once a B preamble has been detected, it is a simple matter of determining whether a decoded subframe relates to channel A or channel B simply by toggling between the channels as subframes are decoded.

[0085]   Moreover, the decoder 100 is able to output an enable pulse indicating that the data signal is valid. For example, the decoder is able to output an enable signal as shown in Fig. 10. Specifically, Fig. 10 shows a decoded data signal 1000 comprising the data bits 100101. The enable signal 1010 includes a short pulse at the commencement of each data bit. Typically, the enable signal will be provided using the free running clock that is generated by the clock generator 10 and drives the sampler 30, the distance counter 50 and the pulse filter 60. Typically, this clock will run at 96MHz and the width of the enable pulse will be a single clock cycle, to allow a sufficiently short enable pulse.

[0086]   Consequently, the decoder 100 of the present invention provides a fully digital decoding solution, which is suitable for implementation with virtually any digital technology, including ASICs, gate arrays and FPGAs.

[0087]   Moreover, no PLL is required to decode the digital stream. Accordingly, where required, the data clock can be recovered after decoding and it is therefore possible to lock on to the broadest range of clock frequencies.

[0088]   A further advantage is that the sampling frequency recovery can be isolated from the data stream decoding. Thus, audio clock recovery and its jitter characteristics can be optimised without cross effects due to data encoding.

[0089]   In addition, the clock requirements of the clock generator 10 are relaxed. All that is required is a clock for sampling the input data stream fast enough that all transitions in the data stream can be detected, different pulse widths can be detected with some degree of safety margin and the widths of different thresholds can be calculated sufficiently accurately from *dmax*. It is thought that at least 4x128fs=512fs will generally be required, although a still faster clock would help to improve the behaviour of the decoder where the incoming IEC

60958 signal already jitters. However, the jitter characteristics and precise frequency of the clock are not important.

[0090]   As noted above, the output of the pulse filter is the fully decoded, serial, digital audio data stream. In typical set ups, the audio clock needs to be recovered in order to process the data without resampling.

[0091]   The audio clock could be recovered using a conventional analogue or digital PLL. However, such arrangements for recovering the audio clock are liable to present the same difficulties as set out above in respect of decoding the coded data stream.

[0092]   Accordingly, the present invention further provides an audio clock recovery arrangement, one embodiment of which is shown in Fig. 11. In this embodiment, the audio clock recovery arrangement 900 comprises a shift register 910, a FIFO 920, a clock control unit 930 and a direct digital synthesis (DDS) unit 940. The DDS unit is a programmable clock that operates by digitally generating a frequency-agile signal from an accurate reference clock. In this embodiment, the DDS generates the audio clock.

[0093]   The shift register 910 is used to create a parallel word, for example for every IEC 60958 (or other) subframe, which is then transferred to the FIFO 920. In other words, as each of the data bits in a subframe or word is received by the shift register 910, it is stored until all the bits in the subframe or word have been received. Once all the bits have been received, they are sent in parallel – that is, at the same time – to the FIFO 920.

[0094]   It should be noted that the shift register 910 can be used to discard any spurious bits output by the decoder, for example, the spurious bits output by the embodiment of the decoder shown in Fig. 9 after respective preambles have been detected.

[0095]   The FIFO 920 is used to store each parallel word for a limited time before it is output to the audio logic block 950 for processing. Thus, several parallel words are stored in the FIFO 920 at any one time before they are sent, successively and in order, to the audio logic block 950.

[0096]   The clock control 930 is provided with a signal indicative of the fill grade of the FIFO 920 – that is, with a measure of how full the FIFO is. The fill grade is used by the clock control 930 to control the frequency of the clock generated by the DDS 940 and hence the speed at which words are used by the audio logic 950. If the clock control 930 detects that the FIFO 920 is beginning to fill up, this is an indication that clock generated by the DDS 940 is too slow compared with the rate of data input to the FIFO. The clock control 930 therefore causes the frequency of the clock to be increased. If, on the other hand, the clock control 930 detects that the FIFO 920 is beginning to empty, this is an indication that clock generated by the DDS 940 is comparatively too fast. The clock control 930 therefore causes the frequency of the clock to be reduced.

[0097]   The required depth of the FIFO 920 is dependent on the jitter requirements of the recovered audio clock. If the jitter requirements of the audio clock are high, then a fast clock is required to reproduce the audio clock. The main reason for this is that when creating a clock by fully digital circuitry, the only event that can cause the audio clock to toggle is a clock edge from the clock generator. In the instant invention, both edges of a 96MHz clock could be used to create a peak jitter of 5.2ns.

[0098]   Of course, the required depth of the FIFO 920 depends on the difference in the data rates of the audio logic and the decoded incoming IEC 60958 signal. Ideally, the data rate of the audio logic is stable. However, the incoming IEC 60958 signal will itself invariably jitter and have a less stable data rate. In addition, the more stable the audio clock, the less its frequency is altered by the FIFO fill grade and the larger the FIFO 920 needs to

be. In other words, the FIFO 320 needs to be able to store enough words at one time so that it is always sufficiently full to allow the audio clock to operate stably. However, a FIFO depth of as little as 16 words should be sufficient for most applications.

[0099] The arrangement shown in Fig. 11 can also be used for a large number of applications other than the recovery of an audio clock. Of course, the logic 950 need not be audio logic. For example, the input to the shift register 910 could be a digital data stream from a microprocessor bus and relate to any of a number of processing applications. Another example would be a data stream from a FireWire application or a data stream from a Bluetooth device. In such data streams, there are variations in the average and peak data rates. With the use of FireWire for audio applications, the average data rate necessarily equals the audio data rate. However, the peak data rate on FireWire is 400Mbps, which is about 250 times the data rate of a standard 16 bit stereo stream. Nonetheless, the arrangement shown in Fig. 11 could accept the FireWire data in the FIFO 920 and still provide a stable clock for use by the logic. Consequently, the arrangement shown in Fig. 9 is effective for re-clocking data streams received at a varied frequency.

[00100] In effect, the arrangement shown in Fig. 9 can act as an elastic buffer for any data signal that passes between different clock domains.

[00101] The foregoing description has been given by way of example only and it will be appreciated by a person skilled in the art that modifications can be made without departing from the spirit and scope of the present invention. For example, the present invention need not be applied only to biphase coded data streams. Rather, it can be applied to other coded data streams, such as triphase coded data streams or other coding systems. The only modifications would be to determine alternative, suitable thresholds and to reconfigure the pulse filter so that its functionality matches the coded data stream and it provides the required output. In addition, the present invention is not limited to audio or video data or to the IEC

60958 or EIAJ CP-340 1987-9 standards, including the AES-EBU and SPDIF implementations. Indeed, aspects of the present invention are applicable to any data stream. Thus, the number of subframes in a block and the number of data bits in a subframe used in any particular scheme are unimportant to the present invention.